Network Working Group   Y. Nir
Internet-Draft   Check Point
Intended status: Standards Track   April 28, 2015
Expires: October 30, 2015

                ChaCha20, Poly1305 and their use in IKE & IPsec
                   draft-ietf-ipsecme-chacha20-poly1305-06

Abstract

   This document describes the use of the ChaCha20 stream cipher along
   with the Poly1305 authenticator, combined into an AEAD algorithm for
   the Internet Key Exchange protocol (IKEv2) and for IPsec.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The Advanced Encryption Standard (AES - [FIPS-197]) has become the
gold standard in encryption. Its efficient design, wide
implementation, and hardware support allow for high performance in
many areas, including IPsec VPNs. On most modern platforms, AES is
anywhere from 4x to 10x as fast as the previous most-used cipher,
3-key Data Encryption Standard (3DES - [SP800-67]). 3DES also has a
64-bit block, which means that the amount of data that can be
encrypted before rekeying is required is not great. These reasons
make AES not only the best choice, but the only choice.

The problem is that if future advances in cryptanalysis reveal a
weakness in AES, VPN users will be in an unenviable position. With
the only other widely supported cipher being the much slower 3DES, it
is not feasible to re-configure IPsec installations away from AES.
[standby-cipher] describes this issue and the need for a standby
cipher in greater detail.

This document proposes the fast and secure ChaCha20 stream cipher as
such a standby cipher in an Authenticated Encryption with Associated
Data (AEAD) construction with the Poly1305 authenticator for use with
the Encapsulated Security Protocol (ESP - [RFC4303]) and the Internet
Key Exchange Protocol (IKEv2 - [RFC7296]). The algorithms are
described in a separate document ([chacha_poly]). This document only
describes the IPsec-specific things.

1.1. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
"SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
document are to be interpreted as described in [RFC2119].

2. ChaCha20 & Poly1305 for ESP

AEAD_CHACHA20_POLY1305 is a combined mode algorithm, or AEAD. The
construction follows the AEAD construction in section 2.8 of
[chacha_poly]:

o The Initialization Vector (IV) is 64-bit, and is used as part of
   the nonce. The IV MUST be unique for each invocation for a
   particular SA but does not need to be unpredictable. The use of a
   counter or a linear feedback shift register (LFSR) is RECOMMENDED.
o A 32-bit Salt is prepended to the 64-bit IV to form the 96-bit
   nonce. The salt is fixed per SA and it is not transmitted as part
   of the ESP packet.
o The encryption key is 256-bit.
o The Internet Key Exchange protocol generates a bitstring called
   KEYMAT using a pseudo-random function (PRF). That KEYMAT is
   divided into keys for encryption, message authentication and
   whatever else is needed. The KEYMAT requested for each
   ChaCha20-Poly1305 key is 36 octets. The first 32 octets are the
   256-bit ChaCha20 key, and the remaining four octets are used as
   the Salt value in the nonce.

The ChaCha20 encryption algorithm requires the following parameters:
a 256-bit key, a 96-bit nonce, and a 32-bit initial block counter.
For ESP we set these as follows:

o The key is set as mentioned above.
o The 96-bit nonce is formed from a concatenation of the 32-bit Salt
   and the 64-bit IV, as described above.
o The Initial Block Counter is set to one (1). The reason that one
   is used for the initial counter rather than zero is that zero is
   reserved for generating the one-time Poly1305 key (see below)

As the ChaCha20 block function is not applied directly to the
plaintext, no padding should be necessary. However, in keeping with
the specification in RFC 4303, the plaintext always has a pad length
octet and a Next Header octet and may require padding bytes so as to
align the buffer to an integral multiple of 4 octets.

The same key and nonce, along with a block counter of zero are passed
to the ChaCha20 block function, and the top 256 bits of the result
are used as the Poly1305 key. The nonce passed to the block function

here is the same nonce that is used in ChaCha20, including the 32-bit
Salt, and the key passed is the same as the encryption key.

Finally, the Poly1305 function is run on the data to be
authenticated, which is, as specified in section 2.8 of [chacha_poly]
a concatenation of the following in the below order:

o The Authenticated Additional Data (AAD) - see Section 2.1.
o Zero-octet padding that rounds the length up to 16 bytes. This is
   4 or 8 bytes depending on the length of the AAD.
o The ciphertext
o Zero octet padding that rounds the total length up to an integral
   multiple of 16 bytes.
o The length of the additional authenticated data (AAD) in octets
   (as a 64-bit little-endian integer).
o The length of the ciphertext in octets (as a 64-bit little-endian
   integer).

The 128-bit output of Poly1305 is used as the tag. All 16 bytes are
included in the packet.

The encryption algorithm transform ID for negotiating this algorithm
in IKE is TBA by IANA.

2.1. AAD Construction

The construction of the Additional Authenticated Data (AAD) is
similar to the one in [RFC4106]. For security associations (SAs)
with 32-bit sequence numbers the AAD is 8 bytes: 4-byte SPI followed
by 4-byte sequence number ordered exactly as it is in the packet.
For SAs with ESN the AAD is 12 bytes: 4-byte SPI followed by an
8-byte sequence number as a 64-bit network order integer.

3. Use in IKEv2

AEAD algorithms can be used in IKE, as described in [RFC5282]. More
specifically:

o The Encrypted Payload is as described in section 3 of that
   document.
o The ChaCha20-Poly1305 keying material is derived similar to ESP:
   36 octets are requested for each of SK_ei and SK_er, of which the
   first 32 form the key and the last 4 form the salt. No octets are
   requested for SK_ai and SK_ar.
o The IV is 64 bits, as described in Section 2, and is included
   explicitly in the Encrypted payload.
o The sender SHOULD include no padding and set the Pad Length field
   to zero. The receiver MUST accept any length of padding.

   o The AAD is as described in section 5.1 of RFC 5282, so it's 32
      bytes (28 for the IKEv2 header + 4 bytes for the encrypted payload
      header) assuming no unencrypted payloads.

4. Negotiation in IKEv2

   When negotiating the ChaCha20-Poly1305 algorithm for use in IKE or
   IPsec, the value xxx (TBA by IANA) should be used in the transform
   substructure of the SA payload as the ENCR (type 1) transform ID. As
   with other AEAD algorithms, INTEG (type 3) transform substructures
   MUST NOT be specified or just one INTEG transform MAY be included
   with value NONE (0).

5. Security Considerations

   The ChaCha20 cipher is designed to provide 256-bit security.

   The Poly1305 authenticator is designed to ensure that forged messages
   are rejected with a probability of $1-(n/(2^{102}))$ for a 16n-byte
   message, even after sending 2^{64} legitimate messages, so it is SUF-
   CMA in the terminology of [AE].

   The most important security consideration in implementing this draft
   is the uniqueness of the nonce used in ChaCha20. The nonce should be
   selected uniquely for a particular key, but unpredictability of the
   nonce is not required. Counters and LFSRs are both acceptable ways
   of generating unique nonces.

   Another issue with implementing these algorithms is avoiding side
   channels. This is trivial for ChaCha20, but requires some care for
   Poly1305. Considerations for implementations of these algorithms are
   in the [chacha_poly] document.

   The Salt value in used nonce construction in ESP and IKEv2 is derived
   from the keystream, same as the encryption key. It is never
   transmitted on the wire, but the security of the algorithm does not
   depend on its secrecy. Thus implementations that keep keys and other
   secret material within some security boundary MAY export the Salt
   from the security boundary. This may be useful if the API provided
   by the library accepts the nonce as parameter rather than the IV.

6. IANA Considerations

   IANA is requested to assign one value from the IKEv2 "Transform Type
   1 - Encryption Algorithm Transform IDs" registry, with name
   ENCR_CHACHA20_POLY1305, and this document as reference for both ESP
   and IKEv2.

7. Acknowledgements

All of the algorithms in this document were designed by D. J.
Bernstein. The AEAD construction was designed by Adam Langley. The
author would also like to thank Adam for helpful comments, as well as
Yaron Sheffer for telling me to write the algorithms draft. Thanks
also to Martin Willi for pointing out the discrepancy with the final
version of the algorithm document, and to Valery Smyslov and Tero
Kivinen for helpful comments on this draft. Thanks to Steve Doyle
and Martin Willi for pointing out mistakes in my examples.

8. References

8.1. Normative References

[RFC2119]   Bradner, S., "Key words for use in RFCs to Indicate
            Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4303]   Kent, S., "IP Encapsulating Security Payload (ESP)", RFC
            4303, December 2005.

[RFC5282]   Black, D. and D. McGrew, "Using Authenticated Encryption
            Algorithms with the Encrypted Payload of the Internet Key
            Exchange version 2 (IKEv2) Protocol", RFC 5282, August
            2008.

[RFC7296]   Kivinen, T., Kaufman, C., Hoffman, P., Nir, Y., and P.
            Eronen, "Internet Key Exchange Protocol Version 2
            (IKEv2)", RFC 7296, October 2014.

[chacha_poly]
            Langley, A. and Y. Nir, "ChaCha20 and Poly1305 for IETF
            protocols", draft-nir-cfrg-chacha20-poly1305-01 (work in
            progress), January 2014.

8.2. Informative References

[AE]   Bellare, M. and C. Namprempre, "Authenticated Encryption:
            Relations among notions and analysis of the generic
            composition paradigm", 2000,
            <http://cseweb.ucsd.edu/~mihir/papers/oem.html>.

[FIPS-197]
            National Institute of Standards and Technology, "Advanced
            Encryption Standard (AES)", FIPS PUB 197, November 2001,
            <http://csrc.nist.gov/publications/fips/fips197/
            fips-197.pdf>.

   [RFC1761] Callaghan, B. and R. Gilligan, "Snoop Version 2 Packet
              Capture File Format", RFC 1761, February 1995,
              <https://tools.ietf.org/html/rfc1761>.

   [RFC4106] Viega, J. and D. McGrew, "The Use of Galois/Counter Mode
              (GCM) in IPsec Encapsulating Security Payload (ESP)", RFC
              4106, June 2005.

   [SP800-67]
              National Institute of Standards and Technology,
              "Recommendation for the Triple Data Encryption Algorithm
              (TDEA) Block Cipher", FIPS SP800-67, January 2012,
              <http://csrc.nist.gov/publications/nistpubs/800-67-Rev1/
              SP-800-67-Rev1.pdf>.

   [standby-cipher]
              McGrew, D., Grieco, A., and Y. Sheffer, "Selection of
              Future Cryptographic Standards", draft-mcgrew-standby-
              cipher (work in progress), January 2013.

Appendix A. ESP Example

   For this example, we will use a tunnel-mode ESP SA using the
   ChaCha20-Poly1305 algorithm. The keying material is as follows:

```
   KEYMAT:
   000  80 81 82 83 84 85 86 87 88 89 8a 8b 8c 8d 8e 8f  ................
   016  90 91 92 93 94 95 96 97 98 99 9a 9b 9c 9d 9e 9f  ................
   032  a0 a1 a2 a3                                       ....
```

   Obviously not a great PRF. The first 32 octets are the key and the
   final four octets (0xa0 0xa1 0xa2 0xa3) are the salt. For the
   packet, we will use an ICMP packet from 198.51.100.5 to 192.0.2.5:

```
   Source Packet:
   000  45 00 00 54 a6 f2 00 00 40 01 e7 78 c6 33 64 05  E..T....@..x.3d.
   016  c0 00 02 05 08 00 5b 7a 3a 08 00 00 55 3b ec 10  ......[z:...U;..
   032  00 07 36 27 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13  ..6'............
   048  14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23  ............ !"#
   064  24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33  $%&'()*+,-./0123
   080  34 35 36 37                                      4567
```

   The SA details are as follows:

   o The key and Salt are as above.
   o The SPI is 0x01 0x02 0x03 0x04.
   o The next sequence number is 5; ESN is not enabled.

   o The gateway IP address for this side is 203.0.113.153; The peer
      address is 203.0.113.5.
   o NAT was not detected.

   The 64-bit IV is 0x10 0x11 0x12 0x13 0x14 0x15 0x16 0x17. Putting
   together the salt and IV we get the nonce:

   The nonce:
   000 a0 a1 a2 a3 10 11 12 13 14 15 16 17   ....

   The plaintext to encrypt consists of the source IP packet plus the
   padding:

  Plaintext (includes padding and pad length):
  000 45 00 00 54 a6 f2 00 00 40 01 e7 78 c6 33 64 05 E..T....@..x.3d.
  016 c0 00 02 05 08 00 5b 7a 3a 08 00 00 55 3b ec 10 ......[z:...U;..
  032 00 07 36 27 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 ..6'............
  048 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 .... !"#
  064 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 $%&'()*+,-./0123
  080 34 35 36 37 01 02 02 04   4567....

   With the key, nonce and plaintext available, we can call the ChaCha20
   function and encrypt the packet, producing the ciphertext:

  Ciphertext:
  000 24 03 94 28 b9 7f 41 7e 3c 13 75 3a 4f 05 08 7b $..(..A~<.u:O..{
  016 67 c3 52 e6 a7 fa b1 b9 82 d4 66 ef 40 7a e5 c6 g.R.......f.@z..
  032 14 ee 80 99 d5 28 44 eb 61 aa 95 df ab 4c 02 f7 .....(D.a....L..
  048 2a a7 1e 7c 4c 4f 64 c9 be fe 2f ac c6 38 e8 f3 *..|LOd.../..8..
  064 cb ec 16 3f ac 46 9b 50 27 73 f6 fb 94 e6 64 da ...?.F.P's....d.
  080 91 65 b8 28 29 f6 41 e0   .e.().A.

   To calculate the tag, we need a one-time Poly1305 key, which we
   calculate by calling the ChaCha20 function again with the same key
   and nonce, but a block count of zero.

  Poly1305 one-time key:
  000 af 1f 41 2c c1 15 ad ce 5e 4d 0e 29 d5 c1 30 bf ..A,....^M.)..0.
  016 46 31 21 0e 0f ef 74 31 c0 45 4f e7 0f d7 c2 d1 F1!...t1.EO.....

   The AAD is constructed by concatenating the SPI to the sequence
   number:

   000 01 02 03 04 00 00 00 05   ....

   The input to the Poly1305 function is constructed by concatenating
   and padding the AAD and ciphertext:

Poly1305 Input:
```
000  01 02 03 04 00 00 00 05 00 00 00 00 00 00 00 00  ................
016  24 03 94 28 b9 7f 41 7e 3c 13 75 3a 4f 05 08 7b  $..(..A~<.u:O..{
032  67 c3 52 e6 a7 fa b1 b9 82 d4 66 ef 40 7a e5 c6  g.R.......f.@z..
048  14 ee 80 99 d5 28 44 eb 61 aa 95 df ab 4c 02 f7  .....(D.a....L..
064  2a a7 1e 7c 4c 4f 64 c9 be fe 2f ac c6 38 e8 f3  *..|LOd.../..8..
080  cb ec 16 3f ac 46 9b 50 27 73 f6 fb 94 e6 64 da  ...?.F.P's....d.
096  91 65 b8 28 29 f6 41 e0 00 00 00 00 00 00 00 00  .e.().A.........
112  08 00 00 00 00 00 00 00 58 00 00 00 00 00 00 00  ........X.......
```

 The resulting tag is:

 Tag:
```
000  76 aa a8 26 6b 7f b0 f7 b1 1b 36 99 07 e1 ad 43  v..&k.....6....C
```

 Putting it all together, the resulting packet is as follows:

 ESP packet:
```
000  45 00 00 8c 23 45 00 00 40 32 de 5b cb 00 71 99  E...#E..@2.[..q.
016  cb 00 71 05 01 02 03 04 00 00 00 05 10 11 12 13  ..q.............
032  14 15 16 17 24 03 94 28 b9 7f 41 7e 3c 13 75 3a  ....$..(..A~<.u:
048  4f 05 08 7b 67 c3 52 e6 a7 fa b1 b9 82 d4 66 ef  O..{g.R.......f.
064  40 7a e5 c6 14 ee 80 99 d5 28 44 eb 61 aa 95 df  @z.......(D.a...
080  ab 4c 02 f7 2a a7 1e 7c 4c 4f 64 c9 be fe 2f ac  .L..*..|LOd.../.
096  c6 38 e8 f3 cb ec 16 3f ac 46 9b 50 27 73 f6 fb  .8.....?.F.P's..
112  94 e6 64 da 91 65 b8 28 29 f6 41 e0 76 aa a8 26  ..d..e.().A.v..&
128  6b 7f b0 f7 b1 1b 36 99 07 e1 ad 43              k.....6....C
```

Appendix B. IKEv2 Example

   For the IKEv2 example, we'll use the following:

   o The key is 0x80..0x9f, the same as in Appendix A.
   o The Salt is 0xa0 0xa1 0xa2 0xa3.
   o The IV will also be the same as in the previous example. The fact
      that the IV and Salt are both the same means that the nonce is
      also the same.
   o Because the key and nonce are the same, so is the one-time
      Poly1305 key.
   o The packet with be an Informational request carrying a single
      payload: A Notify payload with type SET_WINDOW_SIZE, setting the
      window size to 10.
   o iSPI = 0xc0 0xc1 0xc2 0xc3 0xc4 0xc5 0xc6 0xc7.
   o rSPI = 0xd0 0xd1 0xd2 0xd3 0xd4 0xd5 0xd6 0xd7.
   o Message ID shall be 9.

   The Notify Payload:
```
000  00 00 00 0c 00 00 40 01 00 00 00 0a              ......@.....
```

```
Plaintext (with no padding and a zero pad length):
000   00 00 00 0c 00 00 40 01 00 00 00 0a 00         ......@......
```

```
Ciphertext:
000   61 03 94 70 1f 8d 01 7f 7c 12 92 48 89         a..p....|..H.
```

The AAD is constructed by appending the IKE header to the encrypted
payload header. Note that the length field in the IKE header and the
length field in the encrypted payload header have to be calculated
before constructing the AAD:

```
AAD:
000  c0 c1 c2 c3 c4 c5 c6 c7 d0 d1 d2 d3 d4 d5 d6 d7  ................
016  2e 20 25 00 00 00 00 09 00 00 00 45 29 00 00 29  . %........E)..)
```

In this case, the length of the AAD is an integral multiple of 16, so
when constructing the input to Poly1305 there was no need for
padding. The ciphertext is also 16 octets long, so the construction
has no padding at all. Just 32 octets of AAD, 16 octets of
ciphertext, and two 8-octet length fields in little-endian encoding.

```
Poly1305 Input:
000  c0 c1 c2 c3 c4 c5 c6 c7 d0 d1 d2 d3 d4 d5 d6 d7  ................
016  2e 20 25 00 00 00 00 09 00 00 00 45 29 00 00 29  . %........E)..)
032  61 03 94 70 1f 8d 01 7f 7c 12 92 48 89 00 00 00  a..p....|..H....
048  20 00 00 00 00 00 00 00 0d 00 00 00 00 00 00 00   ...............
```

```
Tag:
000   6b 71 bf e2 52 36 ef d7 cd c6 70 66 90 63 15 b2  kq..R6....pf.c..
```

```
Encrypted Payload:
000  29 00 00 29 10 11 12 13 14 15 16 17 61 03 94 70  )..)........a..p
016  1f 8d 01 7f 7c 12 92 48 89 6b 71 bf e2 52 36 ef  ....|..H.kq..R6.
032  d7 cd c6 70 66 90 63 15 b2                       ...pf.c..
```

```
The IKE Message:
000   c0 c1 c2 c3 c4 c5 c6 c7 d0 d1 d2 d3 d4 d5 d6 d7  ................
016   2e 20 25 00 00 00 00 09 00 00 00 45 29 00 00 29  . %........E)..)
032   10 11 12 13 14 15 16 17 61 03 94 70 1f 8d 01 7f  ........a..p....
048   7c 12 92 48 89 6b 71 bf e2 52 36 ef d7 cd c6 70  |..H.kq..R6....p
064   66 90 63 15 b2                                   f.c..
```

The below file in the snoop format [RFC1761] contains three packets:
The first is the ICMP packet from the example in the Appendix A, the
second is the ESP packet from the same appendix, and the third is the
IKEv2 packet from this appendix. To convert this text back into a
file, you can use a Unix command line tools such as "openssl enc -d
-a":

      c25vb3AAAAAAAAACAAAABAAAAGIAAABiAAAAegAAAABVPq8PAAADVdhs6fUQBHgx
      wbcpwggARQAAVKbyAABAAed4xjNkBcAAAgUIAFt6OggAAFU77BAABzYnCAkKCwwN
      Dg8QERITFBUWFxgZGhscHR4fICEiIyQlJicoKSorLC0uLzAxMjM0NTY3AAAAmgAA
      AJoAAACyAAAAAFU+rw8AAAo62Gzp9RAEeDHBtynCCABFAACMI0UAAEAy3lvLAHGZ
      ywBxBQECAwQAAAAFEBESExQVFhckA5QouX9BfjwTdTpPBQh7Z8NS5qf6sbmC1Gbv
      QHrlxhTugJnVKETrYaqV36tMAvcqpx58TE9kyb7+L6zGOOjzy+wWP6xGm1Anc/b7
      lOZk2pFluCgp9kHgdqqoJmt/sPexGzaZB+GtQwAAAG8AAABvAAAAhwAAAABVPq8P
      AAARH9hs6fUQBHgxwbcpwggARQAAYSNFAABAAEd6nywBxmcsAcQUB9AH0AE0IUcDB
      wsPExcbH0NHS09TV1tcuICUAAAAACQAAAEUpAAApEBESExQVFhdhA5RwH40Bf3wS
      kkiJa3G/4lI279fNxnBmkGMVsg==

Author's Address

      Yoav Nir
      Check Point Software Technologies Ltd.
      5 Hasolelim st.
      Tel Aviv 6789735
      Israel

      Email: ynir.ietf@gmail.com